Scala for Professionals

Custom Control Structures

Call By Name vs. Call By Value

By-Name Parameters

- Normally – or rather, by ‘default’ – method parameters are evaluated at the call site.
- Sometimes, however, it makes more sense to *defer* the evaluation to a later point *within the execution of the method*. Some examples include:
  - Avoiding superfluous (or computationally expensive) evaluations such as in `Option.getOrElse()`*
  - When writing Domain Specific Languages (‘DSLs’), to enable Custom Control Abstractions (as we'll cover soon).

---

* Imagine, for example, your ‘default’ value comes from an external cache such as Redis...

By-Name Parameters

Defining Code

We use ⇒ in a type annotation to declare a parameter is evaluated **by-name**:

```
def debug(msg: ⇒ String): Unit =
  if (isDebugEnabled()) println(msg)
```

- By-Name parameters are evaluated *every time* they are used in the method.
- Consider the above example; we save CPU cycles by only evaluating the value of `msg` if debugging is enabled.

Building Custom Control Structures

Custom Control

By combining by-name parameters with currying, we can define methods which look like *built-in control abstractions*

Example: We can temporarily tweak the `OutputStream` used by `println`:

```
withOutput(out) { println("My hovercraft is full of eels!") }
```

Custom Control

This might be a suitable implementation for `withOutput`:

```
def withOutput(out: PrintStream)(block: => Any): Unit = {
  val formerOut = Console.out
  try {
    Console.setOut(out)
    block
  } finally Console.setOut(formerOut)
}
```

- Note that *each time* `block` is referred to, it evaluates dynamically by-name at the call site, rather than statically (a.k.a. by-value) at the function invocation.

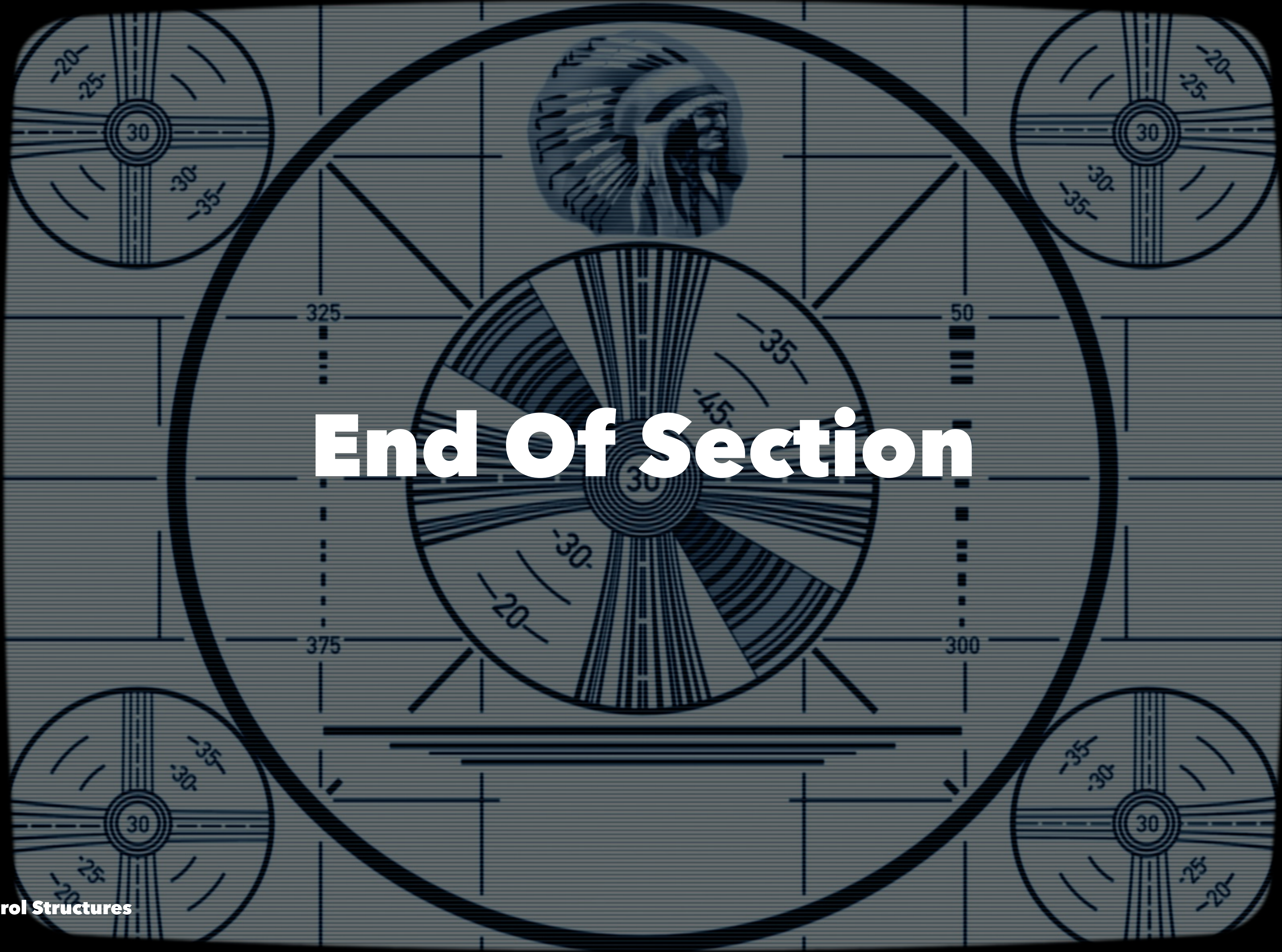
[Optional] Exercise #16: Custom Control Abstractions

- **Attention**: Don't use Scala's `while` for the following
- Create a `repeatWhile` loop which can be used like this:

```
var x = 0
repeatWhile(x < 5) {
  println(x)
  x += 1
}
```

- Get some bonus points by creating a `repeat-until` loop:

```
var x = 0
repeat {
  println(x)
  x += 1
} until(x >= 5)
```



End Of Section